

# Programming Techniques for Supercomputers Tutorial

Erlangen National High Performance Computing Center (NHR@FAU)

Department of Computer Science

FAU Erlangen-Nürnberg

Sommersemester 2026



# Outline

---

- Course website
- Login to the “Fritz” cluster of NHR@FAU
- Starting cluster jobs
- Some guidelines

# Information

---

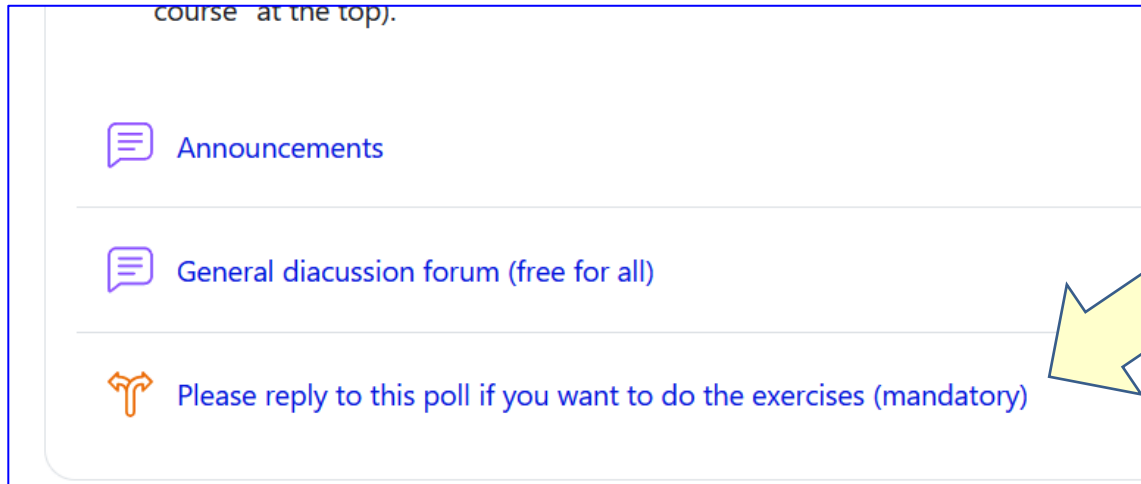
- All slides, exercises, and miscellaneous material can be found on the course pages:

<http://go-nhr.de/PTfS>

- Please use the **discussion forum** there if you have a question that might be of interest to your fellow students
- Please **enroll in the course** so you get all e-mail announcements

# Accounts

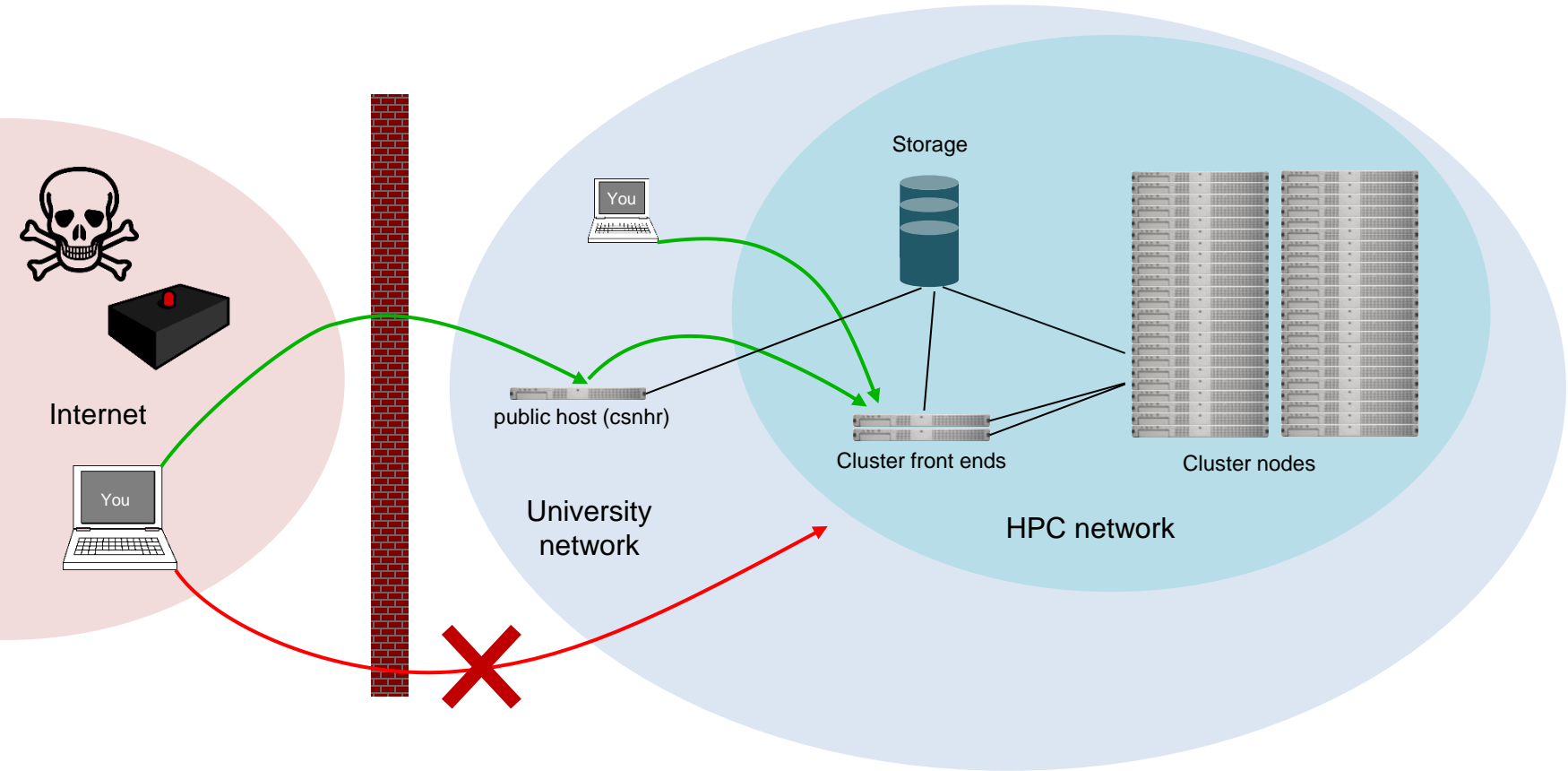
- If you haven't done so already, fill out the poll at:  
<https://moodle.nhr.fau.de/mod/choice/view.php?id=3161>
  - You will get an e-mail invitation about your new account (not right away, we are collecting them and will send them out in batches)



# Accounts (for those who will do the homework)

- **HPC accounts are different** from your IdM accounts
  - **No password** – login is only possible via **SSH public/private key**
  - Manage your HPC account on the **HPC portal**: <https://portal.hpc.fau.de>
    - Log in with your IdM account
  - **Upload an SSH public key** to the portal
    - **Only an uploaded key will grant access!** `~/.ssh/authorized_keys` will be nonfunctional
  - It can take **up to one day** for your account to become active
  - **Video** about generating a key pair under Windows, uploading it to the portal, and logging in: [https://youtu.be/GQ\\_yk\\_P-7Z4](https://youtu.be/GQ_yk_P-7Z4)
    - Video is about MobaXterm under Windows
    - Other options for Windows: WSL, PuTTY, builtin SSH client, Linux VM
- **If you don't know what all this SSH stuff is, learn it. It's a basic skill.**

# NHR@FAU cluster access



# Login to Fritz from inside FAU (or via VPN)

- Login to NHR@FAU cluster front-end machines
  - `ssh ptfsXXXh@fritz.nhr.fau.de`
  - Front-end nodes: `fritz1, ..., fritz4`

```
$ ssh ptfsXXXh@fritz.nhr.fau.de
```

```
=====
Welcome at "Fritz", FAU's NHR and Tier3 parallel computer
- 944 nodes with 2x Intel Icelake 8360Y
[... BLURB ...]
```

```
=====
Last login: Fri Feb  4 14:54:35 2022 from 2001:638:a000:1001::83bc:397
ptfsXXXh@fritz2:~$
```

- Works only from inside FAU (or w/ VPN)  
<https://www.anleitungen.rrze.fau.de/internet-zugang/vpn/>  
(German only 🧑)

# How to log in if **not at the university?**

- Solution 1: Use a VPN (see previous slide)
- Solution 2: Use our “dialog server” **csnhr.nhr.fau.de**
  - All necessary tools installed (Ubuntu 22.04)
  - Access to all HPC systems and most file systems
  - Linux Desktop from Windows: XRDP (think „Windows terminal server“ but with Linux)  
<https://doc.nhr.fau.de/access/xrdp/>
  - **csnhr is not for compiling your code! (see later)**

```
$ ssh ptfsXXXh@csnhr.nhr.fau.de
[...BLURB...]
ptfsXXXh@csnhr:~$ ssh ptfsXXXh@fritz.nhr.fau.de
[...BLURB...]
ptfsXXXh@fritz2:~$
```

# Login via proxy jump

- If you need csnhr only as a “jump host,” there is a faster way to log in
- Add this to your `~/ .ssh/config`:

```
Host fritz
  HostName fritz.nhr.fau.de
  User ptfsXXXh # adapt this
  IdentityFile /home/pi/.ssh/id_rsa # adapt this
  ProxyJump csnhr
Host csnhr
  HostName csnhr.nhr.fau.de
  User ptfsXXXh # adapt this
  IdentityFile /home/pi/.ssh/id_rsa # adapt this
```

- ... and then, just:

```
$ ssh fritz
```

# Work from the Computer Science CIP pools

- CIP Account registration:  
<https://account.cip.cs.fau.de/>
- Start any kind of shell, e.g. **konsole**  
(All workstations in the tutorial room should run linux)
- Login to NHR@FAU cluster front-end machines as usual:
  - **ssh ptfsXXXh@fritz.nhr.fau.de**



# Compiling on Fritz (only on the frontends `fritz*`)

- Make compiler available for use:
  - `module load intel`
  - `icx` → Intel C compiler
  - `icpx` → Intel C++ compiler
  - `ifx` → Intel Fortran compiler
- Recommended Intel compiler options
  - `-O3` high optimization level
  - `-xHost` optimize for CPU the compiler is running on
  - `-fno-alias` assume no overlap between any arrays or elements
  - Other options (`-c`, `-o`, `-S`, `-L`, `-I`, `-l`,...) are the same as for GCC
- Additional software
  - `module available` → overview over all available software
  - `module list` → currently loaded modules
  - `module unload <modulename>` → unload module

# Acquiring a cluster node

- Issue an interactive job (1 node) on Fritz:
  - `salloc --nodes=1 --time=01:00:00`
  - Requests **one full node** for **one hour**
  - Gives you an **interactive login shell** on the compute node
  - For short jobs (< 1h), a node should usually be available right away
- The node is **yours alone** for the allocated time

```
ptfsXXXh@fritz2:~$ salloc --nodes=1 --time=01:00:00
salloc: Pending job allocation 56425
salloc: job 56425 queued and waiting for resources
salloc: job 56425 has been allocated resources
salloc: Granted job allocation 56425
[... BLURB ...]
ptfsXXXh@f0772:~$ ./a.out # this is your program
```

# Fixing the clock frequency

- Modern CPUs can adjust their own clock speed depending on some conditions (“**Turbo Mode**” etc.)
  - # of active cores
  - Temperature
  - ???
- **Accurate and reproducible benchmarking** requires a **constant** clock speed
- Fritz allows you to **set the clock speed when running your binary**
  - Use **srun** with the **--cpu-freq=<MIN>-<MAX>:<governor>** option as a wrapper to your binary
  - Clock frequency is **specified in kHz** here (god knows why...)

```
ptfsXXXh@f0772:~$ srun --cpu-freq=2000000-2000000:performance ./a.out
```



# Job scripts

- Use job scripts for “production runs”
  - Parameter studies
  - Long runs
- Job scripts can be submitted to be executed later when resources are available

Use srun to fix clock frequency to the value specified with `-cpu-freq`

```
#!/bin/bash
#SBATCH --nodes=1 --time=06:00:00
#SBATCH --job-name=TEST01
#SBATCH --export=NONE

unset SLURM_EXPORT_ENV

# the script runs where you submitted it
# do your thing
module load intel
cd ~/PTfS/assignment4
srun ./a.out
```

Block user env from tainting job env

- Options can be specified in the script or on the command line at submission
- Example script: `~ptfs100h/GettingStarted/job.sh`

# Job scripts

- Submit via `sbatch` command, view via `squeue`:

```
ptfsXXXh@fritz2:~$ sbatch job.sh
Submitted batch job 56430
ptfsXXXh@fritz2:~$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
56430	singlenod	TEST01	ptfsXXXh	R	0:01	1	f0767

- After job termination, the stdout and stderr of your job can be found (by default) in a file `<JOBNAME>.o<JOBID>`:

```
ptfsXXXh@fritz2:~$ ls TEST01*
TEST01.o56430
```

# Measuring elapsed time

- Remember: Performance  $P = \frac{W}{T_{wall}}$   
 $W$  = work  
 $T_{wall}$  = “wallclock time,” elapsed time
- Accurate time measurement is important!
  - Very short periods are difficult to measure
  - Measure at least for 100 ms
- Example code in `~ptfs100h/GettingStarted/timing.*`

```
#include "timing.h"
int main(int argc, char *argv[])
{
    double wcTime, wcTimeStart, wcTimeEnd;
    wcTimeStart = getTimeStamp();

    /* PUT YOUR CODE HERE */

    wcTimeEnd = getTimeStamp();
    wcTime = wcTimeEnd - wcTimeStart;
    printf("Walltime: %.3lf s\n", wcTime);

    return 0;
}
```

# General guidelines

---

- **Do not run benchmarks on the frontend nodes**, as multiple programs and users interact there
- You may do test runs, e.g., compilation tests and verification, on frontends
- For obtaining lots of results, **write your own scripts** and execute them via the **batch system**
- **Check your results for plausibility** (**Cool! My code runs @ petaflop/s!**)

# Some links

---

- Linux tutorial for n00bs:
  - <https://ryanstutorials.net/linuxtutorial/>
- MobaXterm SSH client & X server for Windows (choose free version):
  - <https://mobaxterm.mobatek.net/>
- Intel processor details (here for the one used in “Fritz”):
  - [https://en.wikipedia.org/wiki/Ice\\_Lake\\_\(microprocessor\)](https://en.wikipedia.org/wiki/Ice_Lake_(microprocessor))
- Confused about all those CPU code names?
  - <https://en.wikipedia.org/wiki/Xeon>
  - <https://en.wikipedia.org/wiki/Epyc>
- Fritz and Alex cluster official docs:
  - <https://doc.nhr.fau.de/clusters/fritz/>
  - <https://doc.nhr.fau.de/clusters/alex/>

# Assignments

---

- **New homework** assignments are released **Wednesday around noon**
- Report submission
  - **Deadline: eight days later, i.e., Thursday 10:00 a.m. No extensions!**
  - **Deadline for Assignment 0: Thursday, April 30 at 10:00 a.m.**
  - **Upload a single file** in Moodle
    - (searchable) PDF report (**no screenshots!**) or
    - compressed archive including a (searchable) PDF report and supporting material
  - Grading will be done based on PDF report
  - If coding was required, **submit the code** as well!
- Submission allowed in **groups of up to 3** students
  - Everyone still needs to submit on their own
  - **Clearly indicate the partners** in your group

# Tutorial Sessions

---

- Presentation of solution to previous assignment
- Presentation of current (new) assignment
- Opportunity to ask questions

# Report Guidelines

---

- **Report must include**
  - Specific answers to questions/tasks mentioned in assignments
  - Explanation on how you arrived at your answer
  - Description of the steps you took to measure performance/timings/etc.
  - Documentation of compiler switches, frequencies and anything necessary to reproduce your results by someone else (including code, if applicable)
- **Write complete sentences**
  - A part of becoming a scientist is being able to produce intelligible prose
- **Never forget units!**
  - The unit of time is “seconds” or “cycles”
- **When using plots: Label your axes!**
  - **A graph without proper units and scales on the axes will be ignored in grading**